Statements for Database Access

To read data from a database, the following statements are available:

READ	Select a range of records from a database in a specified sequence.
FIND	Select from a database those records which meet a specified search criterion.
HISTOGRAM	Read only the values of one database field, or determine the number of records which meet a specified search criterion.

READ Statement

The following topics are covered:

- Use of READ Statement
- Basic Syntax of READ Statement
- Limiting the Number of Records to be Read
- STARTING/ENDING Clauses
- WHERE Clause
- Further Example of READ Statement

Use of READ Statement

The READ statement is used to read records from a database. The records can be retrieved from the database

- in the order in which they are physically stored in the database (READ IN PHYSICAL SEQUENCE), or
- in the order of Adabas Internal Sequence Numbers (READ BY ISN), or
- in the order of the values of a descriptor field (READ IN LOGICAL SEQUENCE).

In this document, only READ IN LOGICAL SEQUENCE is discussed, as it is the most frequently used form of the READ statement.

For information on the other two options, please refer to the description of the READ statement in the Natural Statements documentation.

Basic Syntax of READ Statement

The basic syntax of the READ statement is:

```
READ view IN LOGICAL SEQUENCE BY descriptor
```

or shorter:

```
READ view LOGICAL BY descriptor
```


view is the name of a view defined in the DEFINE DATA statement (as explained in DEFINE DATA Views).

descriptor is the name of a database field defined in that view. The values of this field determine the order in which the records are read from the database.

If you specify a descriptor, you need not specify the keyword "LOGICAL":

```
READ view BY descriptor
```

If you do not specify a descriptor, the records will be read in the order of values of the field defined as default descriptor (under "Default Sequence") in the DDM. However, if you specify no descriptor, you must specify the keyword"LOGICAL":

```
READ view LOGICAL
```

Example:

```
** Example Program 'READX01'
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 JOB-TITLE
END-DEFINE
READ (6) MYVIEW BY NAME
  DISPLAY NAME PERSONNEL-ID JOB-TITLE
END-READ
END
```

With the READ statement in the above example, records from the EMPLOYEES file are read in alphabetical order of their last names.

The above program will produce the following output, displaying the information of each employee in alphabetical order of the employees' last names:

```
Page      1                                                  99-08-19  13:16:04

        NAME         PERSONNEL          CURRENT
                        ID             POSITION
-------------------- --------- -------------------------

ABELLAN              60008339  MAQUINISTA
ACHIESON             30000231  DATA BASE ADMINISTRATOR
ADAM                 50005800  CHEF DE SERVICE
ADKINSON             20008800  PROGRAMMER
ADKINSON             20009800  DBA
ADKINSON             2001100
```

If you wanted to read the records to create a report with the employees listed in sequential order by date of birth, the appropriate READ statement would be:

```
READ MYVIEW BY BIRTH
```


You can only specify a field which is defined as a "descriptor" in the underlying DDM (it can also be a subdescriptor, superdescriptor or hyperdescriptor).

Limiting the Number of Records to be Read

As shown in the previous example program, you can limit the number of records to be read by specifying a number in parentheses after the keyword READ:

```
READ (6) MYVIEW BY NAME
```

In that example, the READ statement would read no more than 6 records.

Without the limit notation, the above READ statement would read *all* records from the EMPLOYEES file in the order of last names from A to Z.

STARTING/ENDING Clauses

The READ statement also allows you to qualify the selection of records based on the **value** of a descriptor field. With an EQUAL TO/STARTING FROM option in the BY or WITH clause, you can specify the value at which reading should begin. By adding a THRU/ENDING AT option, you can also specify the value in the logical sequence at which reading should end.

For example, if you wanted a list of those employees in the order of job titles starting with "TRAINEE" and continuing on to "Z", you would use one of the following statements:

```
READ MYVIEW WITH JOB-TITLE = 'TRAINEE'
READ MYVIEW WITH JOB-TITLE STARTING from 'TRAINEE'
READ MYVIEW BY JOB-TITLE = 'TRAINEE'
READ MYVIEW BY JOB-TITLE STARTING from 'TRAINEE'
```

Note that the value to the right of the equal sign (=) or STARTING FROM option must be enclosed in apostrophes. If the value is numeric, this *text notation* is not required.

If a BY option is used, a WITH option cannot be used and vice versa.

The sequence of records to be read can be even more closely specified by adding an end limit with a THRU or ENDING AT clause.

To read just the records with the job title "TRAINEE", you would specify:

```
READ MYVIEW BY JOB-TITLE STARTING from 'TRAINEE' THRU 'TRAINEE'
READ MYVIEW WITH JOB-TITLE EQUAL TO 'TRAINEE'
                           ENDING AT 'TRAINEE'
```

To read just the records with job titles that begin with "A" or "B", you would specify:

```
READ MYVIEW BY JOB-TITLE = 'A' THRU 'C'
READ MYVIEW WITH JOB-TITLE STARTING from 'A' ENDING AT 'C'
```

The values are read up to and including the value specified after THRU/ENDING AT. In the two examples above, all records with job titles that begin with "A" or "B" are read; if there were a job title "C", this would also be read, but not the next higher value "CA".

WHERE Clause

The WHERE clause may be used to further qualify which records are to be read.

For instance, if you wanted only those employees with job titles starting from "TRAINEE" who are paid in US currency, you would specify:

```
READ MYVIEW WITH JOB-TITLE = 'TRAINEE'
            WHERE CURR-CODE = 'USD'
```

The WHERE clause can also be used with the BY clause as follows:

```
READ MYVIEW BY NAME
            WHERE SALARY = 20000
```

The WHERE clause differs from the BY/WITH clause in two respects:

- The field specified in the WHERE clause need not be a descriptor.
- The expression following the WHERE option is a logical condition.

The following logical operators are possible in a WHERE clause:

EQUAL	EQ	=
NOT EQUAL TO	NE	¬=
LESS THAN	LT	<
LESS THAN OR EQUAL TO	LE	<=
GREATER THAN	GT	>
GREATER THAN OR EQUAL TO	GE	>=

The following program illustrates the use of the STARTING FROM, ENDING AT and WHERE clauses:

```
** Example Program 'READX02'
DEFINE DATA LOCAL
1 MYEMP VIEW OF EMPLOYEES
  2 NAME
  2 JOB-TITLE
  2 INCOME (1:2)
    3 CURR-CODE
    3 SALARY
    3 BONUS (1:1)
END-DEFINE
*
READ (3) MYVIEW WITH JOB-TITLE = 'TRAINEE' THRU 'TRAINEE'
                WHERE CURR-CODE (*) = 'USD'
  DISPLAY NOTITLE NAME / JOB-TITLE 5X INCOME (1:2)
  SKIP 1
END-READ
END
```

It produces the following output:


```
          NAME                          INCOME
        CURRENT
       POSITION              CURRENCY   ANNUAL     BONUS
                               CODE     SALARY
-------------------------     -------- ---------- ----------

SENKO                         USD           23000          0
TRAINEE                       USD           21800          0

BANGART                       USD           25000          0
TRAINEE                       USD           23000          0

LINCOLN                       USD           24000          0
TRAINEE                       USD           22000          0
```

Further Example of READ Statement

See the following example program in library SYSEXPG:

- READX03

FIND Statement

The following topics are covered:

- Use of FIND Statement
- Basic Syntax of FIND Statement
- Limiting the Number of Records to be Processed
- WHERE Clause
- Example of WHERE Clause
- IF NO RECORDS FOUND Condition
- Example of IF NO RECORDS FOUND Clause
- Further Examples of FIND Statement

Use of FIND Statement

The FIND statement is used to select from a database those records which meet a specified search criterion.

Basic Syntaxof FIND Statement

The basic syntax of the FIND statement is:

```
FIND RECORDS IN view WITH field = value
```

or shorter:

```
FIND view WITH field = value
```

where
view is the name of a view defined in the DEFINE DATA statement (as explained in section DEFINE DATA Views),
field is the name of a database field defined in that view.

You can only specify a *field* which is defined as a "descriptor" in the underlying DDM (it can also be a subdescriptor, superdescriptor, hyperdescriptor or phonetic descriptor).

For the complete syntax, refer to the FIND statement documentation.

Limiting the Number of Records to be Processed

In the same way as with the READ statement, you can limit the number of records to be processed by specifying a number in parentheses after the keyword FIND:

```
FIND (6) RECORDS IN MYVIEW WITH NAME = 'CLEGG'
```

In the above example, only the first 6 records that meet the search criterion would be processed.

Without the limit notation, all records that meet the search criterion would be processed.

Note:
If the FIND statement contains a WHERE clause (see below), records which are rejected as a result of the WHERE clause are **not** counted against the limit.

WHERE Clause

With the WHERE clause of the FIND statement, you can specify an additional selection criterion which is evaluated *after* a record (selected with the WITH clause) has been read and *before* any processing is performed on the record.

Example of WHERE Clause

```
** Example Program 'FINDX01'
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 JOB-TITLE
  2 CITY
END-DEFINE
*
FIND MYVIEW WITH CITY = 'PARIS'
            WHERE JOB-TITLE = 'INGENIEUR COMMERCIAL'
  DISPLAY NOTITLE CITY JOB-TITLE PERSONNEL-ID NAME
END-FIND
END
```

Note that in this example only those records which meet the criteria of the WITH clause *and* the WHERE clause are processed in the DISPLAY statement.


```
      CITY                  CURRENT          PERSONNEL         NAME
                             POSITION             ID
------------------- ------------------------ --------- --------------------

PARIS               INGENIEUR COMMERCIAL      50007300  CAHN
PARIS               INGENIEUR COMMERCIAL      50006500  MAZUY
PARIS               INGENIEUR COMMERCIAL      50004400  VALLY
PARIS               INGENIEUR COMMERCIAL      50002800  BRETON
PARIS               INGENIEUR COMMERCIAL      50001000  GIGLEUX
```

IF NO RECORDS FOUND Condition

If no records are found that meet the search criteria specified in the WITH and WHERE clauses, the statements within the FIND processing loop are not executed (for the previous example, this would mean that the DISPLAY statement would not be executed and consequently no employee data would be displayed).

However, the FIND statement also provides an IF NO RECORDS FOUND clause, which allows you to specify processing you wish to be performed in the case that no records meet the search criteria.

Example of IF NO RECORDS FOUND Clause

```
** Example Program 'FINDX02'
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 NAME
  2 FIRST-NAME
END-DEFINE
*
FIND MYVIEW WITH NAME = 'BLACKMORE'
  IF NO RECORDS FOUND
    WRITE 'NO PERSON FOUND.'
  END-NOREC
  DISPLAY NAME FIRST-NAME
END-FIND
END
```

The above program selects all records in which the field NAME contains the value "BLACKMORE". For each selected record, the name and first name are displayed. If no record with NAME = 'BLACKMORE' is found on the file, the WRITE statement within the IF NO RECORDS FOUND clause is executed:

```
 Page      1                                                  97-08-19  11:44:00

        NAME              FIRST-NAME
-------------------- --------------------

NO PERSON FOUND.
```

Further Examples of FIND Statement

See the following example programs in library SYSEXPG:

- FINDX07
- FINDX08
- FINDX09
- FINDX10
- FINDX11

HISTOGRAM Statement

The following topics are covered:

- Use of HISTOGRAM Statement
- Syntaxof HISTOGRAM Statement
- Limiting the Number of Values to be Read
- STARTING/ENDING Clauses
- WHERE Clause
- Example of HISTOGRAM Statement

Use of HISTOGRAM Statement

The HISTOGRAM statement is used to either read only the values of one database field, or determine the number of records which meet a specified search criterion.

The HISTOGRAM statement does not provide access to any database fields other than the one specified in the HISTOGRAM statement.

Syntax of HISTOGRAM Statement

The basic syntax of the HISTOGRAM statement is:

```
HISTOGRAM VALUE IN view FOR field
```

or shorter:

```
HISTOGRAM view FOR field
```

where
view is the name of a view defined in the DEFINE DATA statement (as explained earlier in section DEFINE DATA Views),
field is the name of the database field defined in that view.

For the complete syntax, refer to the HISTOGRAM statement documentation.

Limiting the Number of Values to be Read

In the same way as with the READ statement, you can limit the number of values to be read by specifying a number in parentheses after the keyword HISTOGRAM:

```
HISTOGRAM (6) MYVIEW FOR NAME
```


In the above example, only the first 6 values of the field NAME would be read.

Without the limit notation, all values would be read.

STARTING/ENDING Clauses

Like the READ statement, the HISTOGRAM statement also provides a STARTING FROM clause and an ENDING AT (or THRU) clause to narrow down the range of values to be read by specifying a starting value and ending value.

Examples:

```
HISTOGRAM MYVIEW FOR NAME STARTING from 'BOUCHARD'
HISTOGRAM MYVIEW FOR NAME STARTING from 'BOUCHARD' ENDING AT 'LANIER'
HISTOGRAM MYVIEW FOR NAME from 'BLOOM' THRU 'ROESER'
```

WHERE Clause

The HISTOGRAM statement also provides a WHERE clause which may be used to specify an additional selection criterion that is evaluated *after* a value has been read and *before* any processing is performed on the value. The field specified in the WHERE clause must be the same as in the main clause of the HISTOGRAM statement.

Example of HISTOGRAM Statement

```
** Example Program 'HISTOX01'
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 CITY
END-DEFINE
*
LIMIT 8
HISTOGRAM MYVIEW CITY STARTING from 'M'
  DISPLAY NOTITLE CITY 'NUMBER OF/PERSONS' *NUMBER *COUNTER
END-HISTOGRAM
END
```

In this program, the system variables *NUMBER and *COUNTER are also evaluated by the HISTOGRAM statement, and output with the DISPLAY statement. *NUMBER contains the number of database records that contain the last value read; *COUNTER contains the total number of values which have been read.

```
          CITY         NUMBER OF   CNT
                       PERSONS
-------------------- --------- ---------

MADISON                      3         1
MADRID                      41         2
MAILLY LE CAMP               1         3
MAMERS                       1         4
MANSFIELD                    4         5
MARSEILLE                    2         6
MATLOCK                      1         7
MELBOURNE                    2         8
```

